A Context-Aware System that Changes Sensor Combinations Considering Energy Consumption

Kazuya Murao[1], Tsutomu Terada[2], Yoshinari Takegawa[3], and Shojiro Nishio[1]

[1] Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita-shi, Osaka 565-0871, Japan
[2] Graduate School of Engineering, Kobe University
1-1 Rokkodai, Nada, Kobe, Hyogo, 657-8501, Japan
[3] Organization of Advanced Science and Technology, Kobe University
1-1 Rokkodai, Nada, Kobe, Hyogo, 657-8501, Japan
{murao.kazuya,nishio}@ist.osaka-u.ac.jp,
{tsutomu,take}@eedept.kobe-u.ac.jp
http://www-nishio.ist.osaka-u.ac.jp/index-e.html

Abstract. In wearable computing environments, a wearable computer runs various applications using various sensors (wearable sensors). In the area of context awareness, though various systems using accelerometers have been proposed to recognize very minute motions and states, energy consumption was not taken into consideration. We propose a context-aware system that reduces energy consumption. In life, the granularity of required contexts differs according to the situation. Therefore, the proposed system changes the granularity of cognitive contexts of a user's situation and supplies power on the basis of the optimal sensor combination. Higher accuracy is achieved with fewer sensors. In addition, in proportion to the remainder of power resources, the proposed system reduces the number of sensors within the tolerance of accuracy. Moreover, the accuracy is improved by considering context transition. Even if the number of sensors changes, no extra classifiers or training data are required because the data for shutting off sensors is complemented by our proposed algorithm. By using our system, power consumption can be reduced without large losses in accuracy.

Keywords: Wearable computing, wearable sensors, context awareness, power consumption.

1 Introduction

The downsizing of computers has led to wearable computing attracting a great deal of attention. Wearable computing is different from conventional computing in three ways[1]: (1) Hands-free operation: information can be obtained without manual operation because the computer is worn. (2) Power always on: the computer is always available because the power is always on. (3) Daily-life support: daily activities can be supported because the computer is worn all the time. Along with the progress in wearable computing, recently, many context-aware systems with various kinds of sensors have been introduced, such as systems with an electromyograph [2], electrocardiogram [3], GSR (Galvanic Skin Reflex) [4], and hand-made devices [5]. In particular, one of

J. Indulska et al. (Eds.): Pervasive 2008, LNCS 5013, pp. 197–212, 2008.
© Springer-Verlag Berlin Heidelberg 2008

the purposes in the Porcupine project[5] is reduction in power consumption. A switch-ball device takes the place of an accelerometer. One switch ball outputs binary data depending on whether it is tilted or not, and nine switch balls go in all directions. Power consumption is very low because of its simplicity, but the accuracy is significantly inferior to that of an accelerometer. This is because an accelerometer has better resolution than that of other devices.

Context-aware systems are applied to many services: health care[4], recognition of workers' routine activity[6], and support of assembly and maintenance tasks[7]. A health-care system[4] recognizes situations of life habits in real time using a heat sensor, GSR sensor, accelerometer, electric sphygmograph, GPS, geomagnetic sensor, and gyroscope. The system recognizes contexts and advises the user about how to make improvements in one's life.

A nurse's routine activity recognition system[6] supports his/her routine work. They have to memorize what they did in a day to communicate with each other and not make a mistake such as giving a dose of medicine needlessly. However, the system seems messy and mistakes might occur. This system recognizes nurses' activities with an accelerometer and their locations with RF-ID receivers.

In the above examples, the accelerometer plays an important role. We consider that the accelerometer is best among current sensors for recognizing behavioral contexts, but the architectures for using accelerometers are not optimal, especially in terms of power consumption. Though the number of sensors in conventional systems are predetermined and fixed, if some sensors can be turned off flexibly, that leads to a reduction in power consumption without much deterioration in accuracy.

In this paper, we propose a context-aware system that changes the combination of accelerometers considering energy consumption. Previously, we have developed the CLAD (cross-linkage for assembled devices) device, which is a relay device between wearable sensors and a wearable computer. CLAD manages the power supply to the sensors[8]. By utilizing CLAD, the proposed system can manage sensors to achieve a high accuracy of activity recognition with a low energy consumption.

This paper is organized as follows. Section 2 describes advanced research contributing to this system. Section 3 presents the system structure. The performance of our system is discussed in Section 4. Finally, Section 5 concludes our research.

2 CLAD

We have proposed CLAD[8] which is a relay device positioned between a wearable computer and wearable sensors. CLAD manages the connected sensors to achieve (1) flexible power supply control for energy saving, and (2) flexible error control for achieving sufficient sensed-data accuracy. The CLAD prototype is shown in Figure 1. The size of CLAD is W76 $\times$ H13 $\times$ D70 mm, and the size of the sensor is W45 $\times$ H12 $\times$ D12 mm.

CLAD has its own power source and manages connected sensors. The voltage and current to detect power shortages and overcurrents are monitored . Each sensor has a microcomputer (CPU) to process commands from CLAD. Information about the sensor (type, accuracy, output range, start-up time, operating voltage, and operating current) is stored in the CPU. CLAD has the following characteristics.

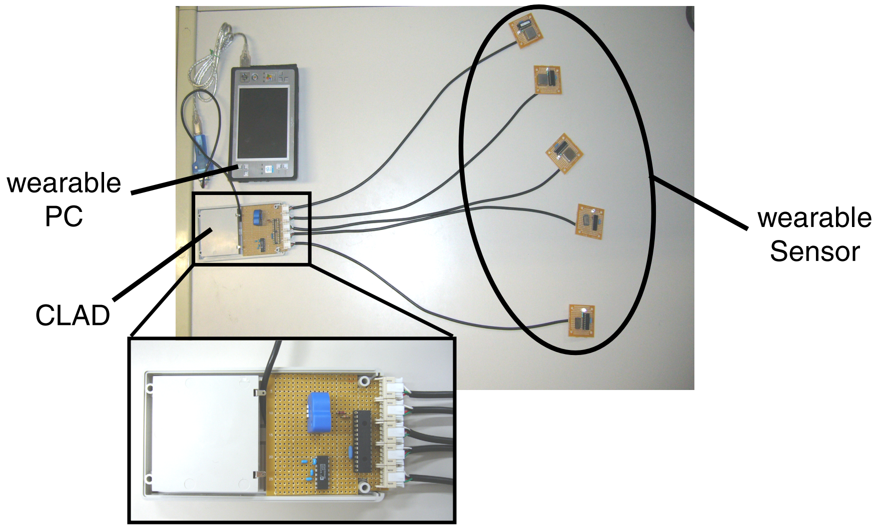


Fig. 1. CLAD prototype

- **Alternative device retrieval and changeover**
  CLAD detects sensor anomalies from consecutive outlying data points and sensor data interruptions, for example. In such cases, CLAD identifies an alternative device by referring to the sensor profile information, and CLAD activates it.
- **Power-supply control**
  CLAD always monitors its internal power source. If CLAD detects a power shortage, power consumption is reduced by stopping the power supply to some of the sensors on the basis of a user-defined policy.
- **Overcurrent detection**
  If an overcurrent is detected, CLAD stops all power supplies for safety.
- **Error detection**
  CLAD detects problems such as outlying data and dying sensor batteries. CLAD notifies the PC of such problems, so applications can deal with them individually such as by displaying a message recommending a battery change.
- **Pseudo data generation**
  When a sensor is turned off and there is no alternative device, CLAD generates pseudo data from learned data and the correlation to other sensors. This function improves operational reliability.

The most distinctive function of CLAD is the pseudo data generation. Generally, there are three answers in response to missing data.

- Listwise deletion
  No sensed data is used when at least one piece of missing data is included. In our assumption, a data complementation is used in case a sensor has broken down. This method cannot be used because missing data comes consecutively in that situation.

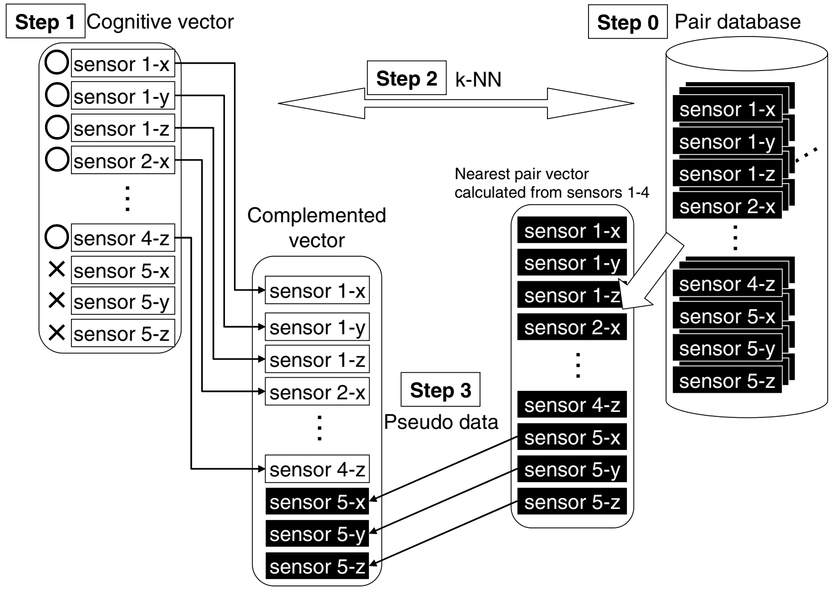


Fig. 2. Pseudo data generation

– Pairwise deletion
  Sensed data is used after removing only missing data. However, a change in sensed data dimension is caused, which requires several restrictions on context recognition algorithms. Therefore, this is not an appropriate answer as a generalized mechanism.
– Imputation
  Sensed data is used after complementing missing data with other values. Doing that does not change the dimension of sensed data, so the user of CLAD need not consider the data complementation.

Considering these characteristics, CLAD uses the imputation for data complementation. An example of pseudo data generation for a context-aware system with five accelerometers is shown in Figure 2. This example supposes that sensor 5 is shut off by a breakdown. The pseudo data is generated as follows.

Step 0. Construct pair database
CLAD has already collected sensed data (pair vectors) for all contexts and constructed a database of pair vectors (pair database).

Step 1. Acquire cognitive vector
When the input contains missing data for some reason such as sensor breakdown, the missing data is removed, and we call the remaining data a cognitive vector.

Step 2. Extract pair vector from pair database
The system finds the pair vector in the database that is nearest the cognitive vector by using the k-NN (k-nearest neighbor) method.

Step 3. Extract pseudo data from pair vector
The data for sensor 5 (missing data) is replaced with that of the extracted pair vector. Then, a complemented vector is generated and will be used as input of a context-aware system.

In pseudo data generation, the distance is calculated among working sensors between cognitive vector $\boldsymbol{X} = (x_{1x}, x_{1y}, x_{1z}, \cdots, x_j, \cdots, x_{5x}, x_{5y}, x_{5z})$ and pair vector $\boldsymbol{P}_i = (p_{i_{1x}}, p_{i_{1y}}, p_{i_{1z}}, \cdots, p_{i_j}, \cdots, p_{i_{5x}}, p_{i_{5y}}, p_{i_{5z}})$ $(i = 1, \cdots, N)$. The subscripts: $1x$ or $5y$ indicate x-axis of sensor 1 or y-axis of sensor 5, and each component such as x_{1x} is a scalar value. N is the number of samples in the pair database. If our mechanism uses Euclidean distance for calculating the distance between a cognitive vector and pair vector, classifying the contexts when the data of working sensors for two contexts are nearly equal and only missing data differs is difficult. Therefore, we focus on the correlation among worn sensor values, and the k-NN method achieves more accurate data complementation by using the correlation. We use the Pearson product-moment correlation coefficient:

$$correlation(x, y) = \left| \frac{\sum_{i=1}^{N}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \overline{x})^2 \sum_{i=1}^{N}(y_i - \overline{y})^2}} \right|, (x \neq y).$$

Generally, an absolute value of 0.0 - 0.2 for the correlation coefficient means there is scarcely any correlation, 0.2 - 0.4 means some correlation, 0.4 - 0.7 means good correlation, and 0.7 - 1.0 means strong correlation.

This method applies the k-NN method to all working sensors and uses the sum of the Euclidean distance divided by the correlation coefficient defined as correlated distance d. The correlation is calculated from variance of the pair data. Data of sensor m is complemented with data of sensor m in the pair vector whose $d_{m,i}$ is minimum, as the following equation shows.

$$d_{m,i} = \sqrt{\sum_{j \in working} \frac{\{x_j - p_{i_j}\}^2}{correlation(x_m, x_j)}}$$

In this method, Euclidean distances among strongly correlated sensors carry much weight and scarcely correlated sensors carry little weight. At last, we find the nearest pair vector $\boldsymbol{P}_{I=argmin_i(d_i)}$, and the system outputs the complemented cognitive vector $\boldsymbol{C} = (c_{1x}, c_{1y}, c_{1z}, \cdots, c_j, \cdots, c_{5x}, c_{5y}, c_{5z})$.

$$c_j = \begin{cases} x_j & (j \in working) \\ p_{I_j} & (j \in malfunctioning) \end{cases}$$

Someone might think that using multiple classifiers for each sensor combination is as practical as our approach. However, an advantage of our approach is that it works independently of classifier. The data of a classifier is always assumed to be complete, and a classifier does not require any configurations. If a better classifier is found in the future, integrating it with our proposal would be easy.

3 System Structure

The purpose of pseudo data generation is to manage hardware errors of sensors (missing data) to maintain the accuracy of context recognition. On the other hand, even when no sensor breaks down, power consumption can be reduced by turning off redundant sensors. Therefore, we focus on the event when cognitive context (context to be recognized) and required accuracy level differ according to situations and applications. We propose a context-aware system, which achieves low battery consumption by considering the situation. In this section, we describe the details of our system and how to reduce power consumption on the basis of the situation. Please note that we have already published a paper on pseudo data generation in [8]. When multiple sensors are loaded for a context-aware system, a result in [8] has demonstrated that unnecessary sensors appear thanks to pseudo data generation. This paper has constructed it as a system. In addition, that we consider the followings for a better contribution.

3.1 Required-Accuracy-Based Power Saving

Required accuracy is different according to the situation. For example, while the highest accuracy is always required in fine-grained services, some users prefer low power consumption (long battery lifetime) in daily activities. In detail, we set a threshold of accuracy. In a serious situation (aerospace, battlefield), we set the accuracy at 90%. Then, the best sensor combination is the least number of sensors needed to satisfy the threshold. On the other hand, in a normal situation, the threshold is set at a lower value. In this way, setting a threshold, we flexibly arranged the trade-off between accuracy and power consumption compared to how the conventional system would have worked only at full power.

However, turning off sensors simply leads to low power consumption and low accuracy. Hence, subsequently, we propose mechanisms to reduce power consumption while maintaining accuracy.

3.2 Context-Granularity-Based Power Saving

Conventional context-aware systems require many sensors to recognize contexts with high accuracy. However, in life, not all trained contexts will be a choice. In detail, while a health-care system needs to recognize many detailed contexts, an information-presentation system on an HMD (Head Mounted Display)[9] just has to judge whether there is movement. Recognizing such easy contexts with fewer sensors is possible.

In this paper, we define *context group* which is a subset of trained contexts. For example, given situations shown in Figure 3, Situation 1 is used in an application that needs to know whether the user is moving. When a user is forbidden by a doctor to exercise strenuously, Situation 2 is used for an application to alert the user in case of high levels of activity. Besides, Situation 3 is used for a health-care application to calculate calorie consumption by recognizing detailed contexts. This method works as follows. First, a user selects a situation according to his/her circumstances or active applications. Second, our system finds the sensor combination whose number of active sensors is least while fulfilling the threshold of accuracy in the same manner as that described in

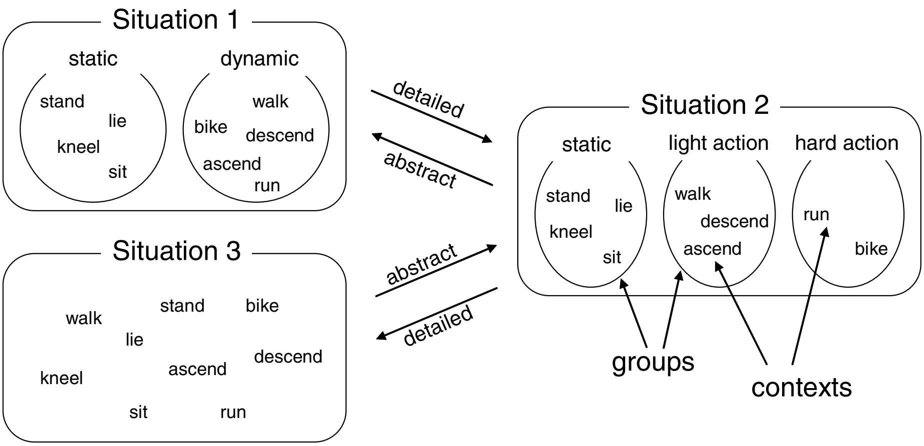


Fig. 3. Context groups

Table 1. Context transitions

Previous context	Possible context
walk	walk, run, stairs, bike, lie, kneel, sit, stand (all contexts)
run	walk, run, stairs, bike, stand
stairs	walk, run, stairs, stand
bike	walk, bike, stand
lie	walk, lie, kneel, sit, stand
kneel	walk, lie, kneel, sit, stand
sit	walk, lie, kneel, sit, stand
stand	walk, run, stairs, bike, lie, kneel, sit, stand (all contexts)

Section 3.1. If there is no situation that the user needs, he/she can define a new situation by regroup contexts. For example, when a user wants to know the context of whether another user is dead, he/she makes *lying*, *kneeling*, and *sitting* a group and makes other contexts another group. When the former context-group lasts for a long time, you may judge if a user is in a critical condition. Judging life and death plays an important role in a wearable system. Detecting death is efficient for military purposes and elderly citizens living alone. Using only one sensor to recognize contexts in Situation 1 is sufficient. By turning redundant sensors off and by complementing data for them, we achieve a low-power-consuming context-aware system with any classifier and training data.

3.3 Context-Transition-Based Power Saving

Focusing on transitions in a person's actions, people basically continue the current context, and that restricts the next context that occurs. A context transition from the result of our preliminary evaluation by 5 people (3 men and 2 women) with 9 contexts (*walking*, *running*, *descending steps*, *ascending steps*, *bicycling*, *lying*, *kneeling*, *sitting,* and *standing*) is shown in Table 1. From Table 1, the candidates of context after *bicycling* are expected to be "go on riding a bike" or "get off a bike". *Lying* and *kneeling* do not happen often in life. For using these characteristics first, we list context candidates from all the contexts shown in Table 1. Second, a classifier is trained for all previous contexts. When a user trains a classifier for recognizing *walking*, training data includes only *walking*, *running*, *stairs*, and *standing* (see Table 1). The other contexts are trained likewise. Finally, when a user is *bicycling*, the system recognizes contexts using the trained classifier for *bicycling*. In this way, restricting candidates of possible contexts based on the current context achieves a high recognition accuracy. This means that our system requires fewer sensors, and power consumption can be reduced. When the number of cognitive contexts have increased, this mechanism becomes more effective. In addition, context transitions are automatically constructed by using a record of daily activities with sensors on at full power.

3.4 Algorithms for Context Recognition

There have been many kinds of algorithms for recognition. Our context-aware system uses Support Vector Machine (SVM)[10] as a classifier. We also implemented several classifiers such as Memory Based Reasoning (MBR) and Self-Organizing Maps (SOMs)[11]. The tendency of evaluation results is the same among all classifiers, and SVM achieves the best total accuracy among them, so we use SVM for the explanation.

SVM is a classification algorithm that often provides competitive or superior accuracy for a large variety of real-world classification tasks[10]. Consider the problem of separating a set of training data $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_J, y_J)$ into two classes, where $\boldsymbol{x}_i \in R^N$ is a feature vector and $y_i \in \{-1, +1\}$ is its class label. Supposing that the classes can be separated by the hyperplane $\boldsymbol{w} \cdot \boldsymbol{x}_i + b$ and no knowledge about the data distribution is given beforehand, the optimal hyperplane is the one with the maximum distance to the closest points in the training dataset. We can find the optimal values for w and b by solving the following problem:

$$\min \frac{1}{2} ||\boldsymbol{w}||^2$$
$$\text{subject to } \; y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1, \; \forall i = 1, \cdots, n.$$

The factor of 1/2 is used for mathematical convenience. By using Lagrange multipliers $\lambda_i (i = 1, \cdots, n)$, the expression is rewritten in this way:

$$\max \sum_{i=1}^{N} \lambda_i - \sum_{i,j=1}^{N} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j, \qquad \text{subject to } \sum_{i=1}^{N} y_i \alpha_i = 0, \; \lambda_i \geq 0.$$

That results in a classification function

$$f(\boldsymbol{x}) = sign\left(\sum_{i=1}^{n} \lambda_i y_i \boldsymbol{x}_i \cdot \boldsymbol{x} + b\right). \tag{1}$$

Most of the λ_i take the value zero. Those $f(\boldsymbol{x_i})$ with nonzero λ_i are so-called support vectors, all of which are on each hyperplane. In cases where the classes are not separable, the Lagrange multipliers are modified to $0 \leq \lambda_i \leq C, i = 1, \cdots, n$, where C is the penalty for misjudgement. This arrangement is called *soft margin* and is the reason SVM performs well.

The original optimal hyperplane algorithm proposed by Vapnik was a linear classifier. To obtain a nonlinear classifier, one maps the data from the input space $\mathbb{R}^N$ to a high dimensional feature space by using $\boldsymbol{x} \to \Phi(\boldsymbol{x})$. However nonlinear classifiers were created by applying the kernel trick to maximum-margin hyperplanes. Assuming there exists a kernel function $K(\boldsymbol{x}, \boldsymbol{x}') = \Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x}')$, a nonlinear SVM can be constructed by replacing the inner product $\boldsymbol{x} \cdot \boldsymbol{x}'$ by the kernel function $K(\boldsymbol{x}, \boldsymbol{x}')$ in Eq. 1. Commonly used kernels are polynomials $K(\boldsymbol{x}, \boldsymbol{x}') = (\gamma \boldsymbol{x} \cdot \boldsymbol{x}' + c)^d$, the Gaussian Radial Basis Function (RBF) $K(\boldsymbol{x}, \boldsymbol{x}') = exp(-\gamma ||\boldsymbol{x} - \boldsymbol{x}'||^2)$, and the sigmoid $K(\boldsymbol{x}, \boldsymbol{x}') = tanh(\gamma \boldsymbol{x} \cdot \boldsymbol{x}' + c)$.

We have examined the kernels while changing their parameters: penalty C: 5,000, 50,000, and 500,000; γ in RBF and sigmoid: 0.0001, 0.005, 0.001, 0.01, 0.1, and 1; and constant c in RBF and sigmoid: 0, 0.1, and 1. No kernel exhibited better performance than that of linear classification, and a C of 50,000 exhibited the best performance. The extension of a 2-class SVM to the N-class can be achieved, e.g., by training N SVMs, one class will be separated from the others.

4 Evaluation

In this section, we evaluate our system on the basis of accuracy and power consumption.

4.1 Evaluation Environment

To evaluate our system, training data and test data were captured by five different test subjects who wore five sensors: both wrists, both ankles, and hip. They acted according to the scenario shown in Table 2. Each instruction is very simple. Instructions have a high degree of freedom in activity, such as stopping halfway or walking to talk to other people. This scenario includes the following nine basic activities: *walking*, *running*, *ascending steps*, *descending steps*, *bicycling*, *lying*, *kneeling*, *sitting*, and *standing*[5]. The former four activities are dynamic and the latter five are static. The worn sensors were three-axis accelerometers[12]. The sampling frequency was 20 Hz. The algorithm for context awareness is Support Vector Machine (SVM) described in Section 3.4. Raw data and hand-labeled contexts of two test subjects in the scenario are shown in Figure 4. As shown in the figure, though general actions are similar to each other, detailed actions are different. The subject of the upper part of the graph in the figure sometimes stops while walking. On the other hand, there is little change in contexts for the subject of the lower graph. In addition, before riding on a bicycle, the subject in the upper graph stands, and the subject in the lower graph walks. In this way, the data used in a evaluation contains various characteristics, so this data is suited for the evaluation.

Generally, using a context-aware algorithm, raw data would not be used but preprocessed for extracting the feature values to grasp the meaning of sensed data. Supposing time $t = T$ now, the constructed context-aware system uses mean $\mu_i(T)$ and variance $\sigma_i(T)$ for 20 samples of 15-dimensional sensed data (cognitive vector) $c_i(T)$ $(i = 1, \cdots, 15)$ retraced from time $t = T$.

$$\mu_i(T) = \frac{1}{20} \sum_{t=T-19}^{T} c_i(t)$$

$$\sigma_i(T) = \frac{1}{20} \sum_{t=T-19}^{T} \Big\{ c_i(t) - \mu_i(t) \Big\}^2$$

Characteristic vector $\boldsymbol{Z}(T)$ is normalized using the following equation for 30-dimensional vector $\boldsymbol{X}(T) = [\mu_1(T), \cdots \mu_{15}(T), \sigma(T) \cdots \sigma_{15}(T)]$, where $\boldsymbol{M}$ and $\boldsymbol{S}$ are the mean and the standard deviation of $\boldsymbol{X}$, respectively.

Table 2. Scenario performed in evaluation

Outdoor phase	**Instruction: Go to buy a juice at the co-op by bicycle** Laboratory → down stairs → to bicycle shed through corridor → to co-op by bicycle → buy juice from a vending machine → back to the lab.
Indoor phase	**Instruction: Read a journal and rest. Then, go upstairs for a job** look for a journal on bookshelves → read the journal on a chair → take a rest on a sofa → recall a job and run upstairs → back to the lab.

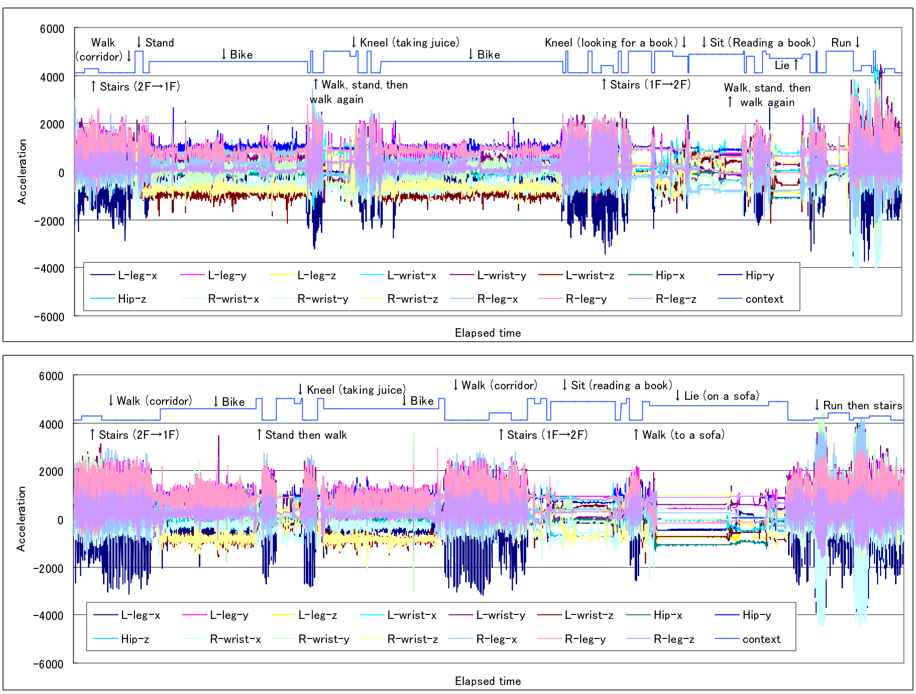


Fig. 4. Raw data and hand-labeled context of test subjects

Table 3. Power consumption @ 5.18 V

Hardware	Power consumption $[mW]$
CLAD only	92.204
Inactive sensor	11.396
Active sensor	40.922

$$\boldsymbol{Z}(T) = \frac{\boldsymbol{X}(T) - \boldsymbol{M}}{\boldsymbol{S}}$$

After this conversion, the mean and variance of $\boldsymbol{Z}(T)$ become 0 and 1, respectively.

The logged data in the scenario were manually labeled, 20% of which becomes training data and the data for complementing while the remaining 80% of the data is used for testing. The amount of the data used for complementing is much less than that in testing, so our proposal makes a significant contribution without using all possible data sets of all remaining sensors. In addition, the pair database is easy to construct because its data need not be labeled.

4.2 Results

First, we measured power consumption of our hardware: CLAD and sensors. The results are shown in Table 3. Each inactive sensor consumes 11.4 mW as a standby power requirement. "CLAD only" means the power consumption for CLAD itself without any sensor. According to this table, 297 mW are consumed in full-power operation (5 active sensors and CLAD).

Evaluation of Context Group. The first result is the accuracy of the context groups described in Section 3.2. The results are plotted in the group $\times$ group confusion matrices shown in Figure 5. These results were obtained with five active sensors and without any complementing. Each cell indicates the number of positives per activity (with the true positives diagonally), the accuracy indicates the percentage of true positives over each activity. The matrix makes the difficulty of each context clear: which context is easily recognizable. As you see, the accuracies are vastly different by a context: *bicycling* and *lying* are high, but *descending* and *kneeling* are low. For this result, the clear point is that a more abstract group achieves a better classification percentage.

As a second result, the accuracy in changing the complementing method for each context-group is plotted in Figure 6. The horizontal axis indicates 31 combinations of active and inactive sensors (◯ means active, a blank means inactive). The vertical axis indicates the accuracy of context recognition. The partitions in the graph indicate a border between active sensors. As mentioned above, the more abstract a situation is, the more the accuracy increases. As sensors are turned off, the accuracies decreases, but their decreases are small due to the complementing mechanism, as described in Section 2. For a comparison, we show the accuracy without our complementing. In this case, inactive sensor data is replaced with an average of the other active sensor data. If not complemented well, the decreases are significant[8]. For this result, in Situation 1 with more than one sensor, the accuracies are the same as that at full power. Situation 2

			Accuracy
Static	22561	2339	90.61%
Dynamic	2564	53835	95.28%

(a) Situation 1: 92.94% on average (not over all data but contexts)

				Accuracy
Static	22346	1675	879	89.75%
Light action	1571	23694	673	91.35%
Hard action	474	821	32707	96.19%

(b) Situation 2: 92.43% on average (not over all data but contexts)

										Accuracy
Walking	17513	75	102	155	308	12	68	147	1578	87.75%
Running	132	2001	118	98	0	0	0	0	13	84.72%
Descending	580	13	2340	20	0	0	0	0	46	78.02%
Ascending	489	14	15	2457	5	0	0	0	0	82.45%
Bicycling	304	4	6	23	31141	0	0	35	127	98.42%
Lying	4	0	0	0	0	5684	0	44	28	98.68%
Kneeling	130	0	0	22	138	0	1841	21	168	79.35%
Sitting	147	0	0	3	35	46	20	7183	147	94.75%
Standing	835	51	23	41	160	0	223	51	7856	85.02%

(c) Situation 3: 87.69% on average (not over all data but contexts)

Fig. 5. Confusion matrices for each situation

exhibits the same tendency as that of Situation 1. Though Situation 3 also has the same tendency, the accuracies on the whole are worse than that of other situations because of the cognitive complexity. In short, the power consumption can be reduced by turning off sensors while maintaining the accuracy, and the accuracy increases with an appropriate situation. Optimal sensor combinations in each situation are shown in Table 4. The tolerances of accuracy are supposed to be 94, 90, and 87%. We assume that the tolerance is the accuracy decided by the user or application; under severe conditions, tolerance will be high, or tolerance may be low for a long battery lifetime in daily life. In each situation with each tolerance, our system selects the fewest number of sensors so that the accuracy of a combination satisfies the tolerance. In the present circumstances, we need to determine an optimal sensor combination for each situation by actual measurement in the same manner as that shown in Figure 6. Power consumption is calculated from Table 3. (e.g., with four active sensors, the power consumption becomes $92.2 + 40.9 \times 4 + 11.4 \simeq 267[mW]$.) The reduction rate is the percentage of power consumption

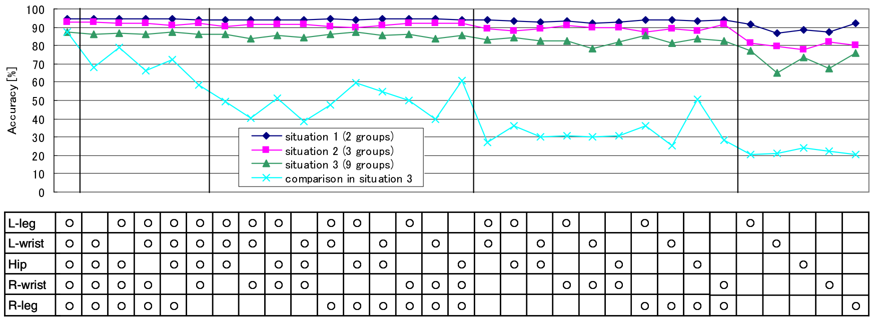


Fig. 6. Accuracy vs. sensor combination in each situation

Table 4. Optimal sensor combinations and their power consumption

Tolerance	Situation	No. of sensors	Combination of active sensors	Accuracy [%]	Power consumption $[mW]$	Reduction [%]
94%	1	3	L-wrist, Hip, R-leg	94.30	238	19.9
	2	5	ALL SENSORS	92.72	297	0
	3	5	ALL SENSORS	87.38	297	0
90%	1	1	R-leg	92.03	179	39.8
	2	2	R-leg, R-wrist	91.50	208	29.8
	3	5	ALL SENSORS	87.38	297	0
87%	1	1	R-leg	92.03	179	39.8
	2	2	R-wrist, R-leg	91.50	208	29.8
	3	3	L-leg, Hip, R-leg	87.08	238	19.9

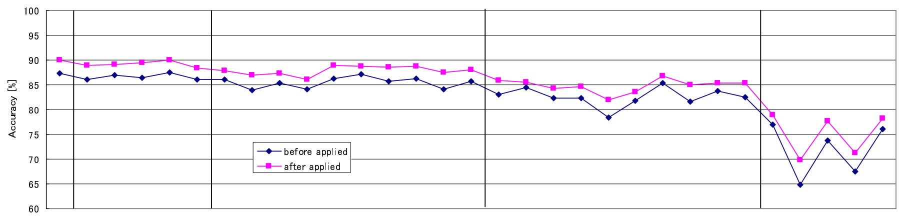
that is reduced compared to that in full power. Note that if no combination achieves the tolerance, the system works with five active sensors. Even with 94% tolerance, power consumption is 20% reduced in Situation 1. Moreover, with 87% tolerance, not all five sensors are not required in all situations.

Evaluation of Context Transition. The accuracy of context recognition considering the human context transition in Table 1 is shown in Table 5. These results were obtained with five active sensors. According to the result, the accuracy was improved for all contexts. For example, when a user changes his/her action from *bicycling* to *walking*, the system made 75 mistakes for *running* and 147 mistakes for *sitting* without considering transition. However, considering the context transition, *running* and *sitting* were excepted from the context candidates. By removing unimaginable contexts, from *bicycling*, *walking* was recognized to be 91.74% (3.99% improved). Accuracies when context transition is applied in all sensor combinations are shown in Figure 7. The environment is Situation 3 (9 contexts). Each axis is the same as that of Figure 6. According to Figure 7, the accuracies in all combinations were improved (2.55% on average). A combination of a smaller number of sensors, which has not been selectable because of low accuracy, becomes selectable, e.g., an accuracy at full power before applying

Table 5. Change in accuracy by context-transition

Previous context	Next context	Accuracy	
		Before	After
run	walk	87.75	90.49
	run	84.72	87.10
	descend	78.02	82.31
	ascend	82.45	84.21
	bicycle	98.42	98.47
	stand	85.02	88.17
stairs	walk	87.75	92.32
	run	84.72	88.03
	descend	78.02	84.35
	ascend	82.45	85.79
	stand	85.02	89.31

Previous context	Next context	Accuracy	
		Before	After
bicycle	walk	87.75	91.74
	bicycle	98.42	98.58
	stand	85.02	88.89
lie kneel sit stand	walk	87.75	93.19
	lie	98.68	99.24
	kneel	79.35	91.14
	sit	94.75	97.53
	stand	85.02	87.52



L-leg	○		○	○	○	○	○	○	○		○	○		○			○	○		○			○				○				
L-wrist	○	○		○	○	○	○	○		○	○		○		○		○		○		○			○				○			
Hip	○	○	○		○	○	○		○	○		○	○			○		○	○			○			○				○		
R-wrist	○	○	○	○		○		○	○	○				○	○	○				○	○	○				○				○	
R-leg	○	○	○	○	○						○	○	○	○	○	○							○	○	○	○					○

Fig. 7. Accuracy vs. sensor combination before and after applying context-transition

context transition (87.38%) is overtaken by that of 3 sensors after applying context transition (88.91%).

Finally, we consider a combination of all mechanisms. A context-granularity-based method and a context-transition-based method can co-exist. Considering the context-transition of context groups in this paper is difficult. Static and Dynamic groups can change to each other in Situation 1. Static, light action, and hard action groups can also change to each other. However, when there are more contexts to recognize, there will be many context groups. In such a case, by using our proposals at the same time, restricting transitions between context groups results in a better performance. Further evaluation and power measurement is part of our future work.

5 Conclusion

We have constructed a context-aware system that changes sensor combinations considering the energy consumption. By assuming the granularity of cognitive contexts differs according to situations, we defined "context group" by including some similar contexts.

In addition, we focused on a transition in human actions to improve the accuracy by reducing the number of the candidates of possible contexts. The proposed system changes the granularity of cognitive contexts of a user's situation and manages the power supply on the basis of an optimal sensor combination. From the evaluation, clearly, not all sensors are needed to recognize required contexts according to situations. As a result, our system has achieved a reduction in energy consumption. The advantage of our system is that even if the number of sensors changes, the system does not require any extra classifiers and training data because the data for sensors that have been shut off is complemented by our proposed algorithm.

As future work, we plan to propose a mechanism for automatic change of the current situation. In our current system, we have to change the situation by hand or another device. The system may be able to decide the current situation by using co-occurrence information among contexts. In addition, context transition is applied according to a binary decision. There is a method limiting context with conditional probability such as in a Bayesian network. Such probabilistic approaches have flexibility, but they have the weakness of unexpected events such as change of context, which rarely happens. This problem is our ongoing study. Furthermore, we think our approach is applicable to wireless sensors that use sleep commands. We also plan to evaluate power consumption of wireless sensors.

Acknowledgements

This research was supported in part by a Grant-in-Aid for Scientific Research (A) (17200006) and Priority Areas (19024046) of the Japanese Ministry of Education, Culture, Sports, Science and Technology, and by the IPA Exploratory Software Project entitled "Development of a Rule Processing Engine for Constructing Wearable Applications."

References

1. Miyamae, M., Terada, T., Tsukamoto, M., Nishio, S.: Design and Implementation of an Extensible Rule Processing System for Wearable Computing. In: Proc. of the 1st IEEE Intl. Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous), pp. 392–400 (August 2004)
2. Toda, M., Akita, J., Sakurazawa, S., Yanagihara, K., Kunita, M., Iwata, K.: Wearable Biomedical Monitoring System Using TextileNet. In: Proc. of the 10th IEEE Intl. Symposium on Wearable Computers (ISWC 2006), pp. 119–120 (October 2006)
3. Shen, C.L., Kao, T., Huang, C.T., Lee, J.H.: Wearable Band Using a Fabric-Based Sensor for Exercise ECG Monitoring. In: Proc. of the 10th IEEE Intl. Symposium on Wearable Computers (ISWC 2006), pp. 143–144 (October 2006)
4. Ouchi, K., Suzuki, T., Doi, M.: LifeMinder: A wearable Healthcare Support System Using Users Context. In: Proc. of the 2nd Intl. Workshop on Smart Appliances and Wearable Computing (IWSAWC 2002), pp. 791–792 (July 2002)
5. Laerhoven, K.V., Gellersen, H.W.: Spine versus Porcupine: a Study in Distributed Wearable Activity Recognition. In: Proc. of the 8th IEEE Intl. Symposium on Wearable Computers (ISWC 2004), pp. 142–149 (October 2004)

6. Naya, F., Ohmura, R., Takayanagi, F., Noma, H., Kogure, K.: Workers Routine Activity Recognition using Body Movement and Location Information. In: Proc. of the 10th IEEE Intl. Symposium on Wearable Computers (ISWC 2006), pp. 105–108 (October 2006)
7. Stiefmeier, T., Ogris, G., Junker, H., Lukowics, P., Tröster, G.: Combining Motion Sensors and Ultrasonic Hands Tracking for Continuous Activity Recognition in a Maintenance Scenario. In: Proc. of the 10th IEEE Intl. Symposium on Wearable Computers (ISWC 2006), pp. 97–104 (October 2006)
8. Murao, K., Takegawa, Y., Terada, T., Nishio, S.: CLAD: a Sensor Management Device for Wearable Computing. In: Proc. of 7th Intl. Workshop on Smart Appliances and Wearable Computing (IWSAWC 2007) (June 2007)
9. Ho, J., Intille, S.S.: Using Context-Aware Computing to Reduce the Perceived Burden of Interruptions from Mobile Devices. In: Proc. Conference on Human Factors in Computing System (CHI 2005), pp. 909–918 (April 2005)
10. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, Heidelberg (1995)
11. Kohonen, T.: Self-Organizing Maps. Springer, Heidelberg (1996)
12. Wirelss Technologies, Inc.: http://www.wireless-t.jp/